Katholieke Universiteit Leuven

Department of Electrical Engineering

ESAT-SISTA/TR 1993-38

A Multi-Objective Optimization Approach for Parameter Setting in System and Control Design[1]

Paul Vanvuchelen[2] and Bart De Moor[3]

June 1992

[1]This report is available by anonymous ftp from This report is available by anonymous ftp from *gate.esat.kuleuven.ac.be* in the directory *pub/SISTA/vanvuche/reports/moopscd.ps.Z*

[2]ESAT - Katholieke Universiteit Leuven, Kardinaal Mercierlaan 94, 3001 Leuven (Heverlee), Belgium, tel 32/16/22 09 31, fax 32/16/22 18 55, email: *paul.vanvuchelen@esat.kuleuven.ac.be* and *bart.demoor@esat.kuleuven.ac.be.* This research was sponsored by the Flemish Institute for Scientific and Technological Research in Industry (I.W.T.) and the Belgian programme on interuniversity attraction poles (IUAP-50) initiated by the Belgian State - Prime Minister's Office - Science Policy Programming. The scientific responsibility is assumed by its authors.

[3]Bart De Moor is a research associate with the N.F.W.O. (Belgian National Fund for Scientific Research).

Abstract

The problem of tuning the free parameters for a fixed structure arises in many engineering applications. Often, different measures of quality are considered simultaneously, so performance trade-offs are inevitable.

In this work, a general framework to handle these problems as multi-objective optimization problems is presented. First, the fixed structure is parameterized, next, the set of design specifications is split into two subsets. The subset of hard specifications determines the feasible parameter region, while the subset of soft specifications represents the performances to be traded-off. It is then relatively straightforward to formulate the problem as a multi-objective optimization problem.

To solve it, different techniques are available in the literature. Depending on the parametrization and on the design specifications, one can determine if global or only local solutions may be expected. For many applications, only local solutions can be guaranteed, because non-convex problems arise. However, experiments show that the obtained solutions are often the global solutions.

1 Introduction

In applications, system-and-control engineers are often confronted with the problem of tuning the free parameters of a fixed structure, while a variety of static and dynamic performance specifications is under consideration.

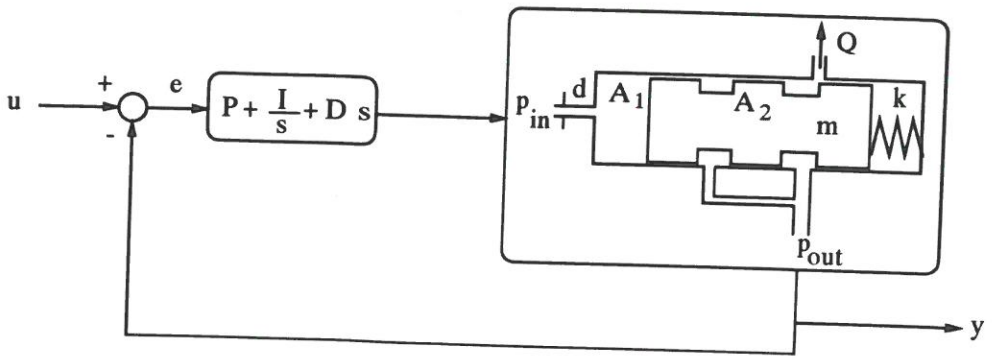


Figure 1: *An electronically controlled hydraulic actuator. First, the mass m, the spring constant k, the surfaces A_1 and A_2 and the diameter d need to be chosen by the system engineer, taking into account specifications as stability, required linearity range, size limits, bandwith and sensitivity. Afterwards, the control engineer must choose P, I and D to meet specifications as stability, tracking, bandwith, sensitivity, settling time and undershoot.*

In figure 1, an electronically controlled hydraulic actuator is shown as an example. First, the system engineer has to assign suitable values to the different physical and geometric parameters of the fixed physical structure. Next, the control engineer has to find proper values for the controller parameters, given the fixed control configuration and the fixed control law. To find a design on the performance boundary (the trade-off boundary), the problem must be handled as a multi-objective optimization problem (MOOP), because heuristic tuning rules will almost never lead to a design on this boundary.
This paper is organized as follows: Section 2 explains how a practical design problem can be translated into a MOOP, while Section 3 contains the theory about MOOPs and treats a class of methods to solve it. A numerical example is given in Section 4. Finally, in Section 5, different ways to reduce the design time are analysed.

2 From real life to a multi-objective optimization problem

Three different phases are needed to transform a practical problem into a MOOP. Although some parts look superfluous from an engineer's point of view, it is good practice to investigate all of them carefully.

2.1 Phase 1: Parametrization

In this phase, the fixed structure is parametrized, which means that the unknowns are chosen. Let $\vec{X} \in \mathbb{R}^n$ be the vector containing the n unknown parameters.

It is the engineer's task to assign a suitable value to all elements x_i of $\vec{X}$. A system designer has to decide upon the values of physical and geometric properties, while a control engineer

has to find values for the unknowns in the control scheme. Different examples can be found in [1], [2], [9] and [20].

2.2 Phase 2: Classifying the dynamical design specifications

2.2.1 Phase 2a: Trivial specifications and quality specifications

When analyzing the different engineering specifications, two main categories arise immediately:

- Trivial specifications, which include the following, in fact trivial, requirements:
  - Realizability: the obtained design may not give rise to contradictions in the design specifications (e.g. when the mass m (x_1) and the volume V (x_2) of an iron cylinder (density ρ fixed) are both considered as design parameters, then the realizability condition $x_1 = \rho x_2$ needs to be taken into account to prevent contradictions in the design specifications).
  - Stability: it is required that the obtained design results in a stable system.
  - Limitations on the parameter size: the design needs to be implementable (e.g. construction limits, number representation limits,...).
- Quality specifications, to guarantee a certain performance level.

  In the literature, many performance measures are proposed to express a system's quality. For dynamical design, examples of performance measures can be found in [2], [15] and [4]. While overshoot and undershoot, rise time, settling time and l_p-norms of various signals, for step responses and polynomial responses, are examples of quality measures for time domain performance, bandwidth, phase margin and gain margin are quality measures for frequency domain performance. Note that all measures are non-negative. Further, it holds that the highest quality, with respect to a certain performance measure, corresponds to the smallest value of that measure.

  In practice, a system's quality is expressed using a quality vector instead of one single quality measure.

2.2.2 Phase 2b: Hard and soft specifications

In this phase, the complete set of specifications, found during phase 2a, is split into two subsets, to allow the designer to formulate the MOOP:

- The set of hard specifications contains all specifications that need to be satisfied to have a feasible design. In practice, it contains the trivial specifications and those quality measures for which no trade-off is allowed by the designer (e.g. the overshoot in the stepresponse is required to be smaller then 10% to be a feasible design).

  Mathematically, this set can be represented by a set of m (in)equalities in $\vec{X}$, determining the set of feasible designs

$$\mathcal{X} = \left\{ \vec{X} \in \mathbb{R}^n \mid \Phi_{hard,i}(\vec{X}) \geq 0,\ i = 1 \ldots m \right\}$$

- The set of soft specifications includes all the performance measures that must be traded-off.

  Mathematically, these k specifications can be combined in the mapping $\vec{\Phi}_{soft}$, whose components $\vec{\Phi}_{soft,j}(\vec{X})$ are functions of $\vec{X}$. In fact, $\vec{\Phi}_{soft}$ maps the parameter space $\mathbb{R}^n$ on the performance space $\mathbb{R}^k$. The image of the set of feasible designs $\mathcal{X}$ corresponds to the set of attainable performances

$$\mathcal{F} = \left\{ \vec{\Phi}_{soft}(\vec{X}) \mid \vec{X} \in \mathcal{X} \right\}$$

A graphical representation for a problem with two unknowns ($n = 2$) and two soft specifications ($k = 2$) is shown in figure 2.

2.3 Phase 3: Formulation as a multi-objective optimization problem

Once the phases 1 and 2 are completed, it is simple to express the designer's wish: minimize all the soft specifications simultaneously, without violating the hard specifications. Mathematically, the problem can be stated as

$$\min_{\vec{X}} \ \vec{\Phi}_{soft}(\vec{X}) \ \ s.t. \ \ \vec{\Phi}_{hard}(\vec{X}) \geq 0$$

or equivalently

$$\min_{\vec{X} \in \mathcal{X}} \ \vec{\Phi}_{soft}(\vec{X})$$

The next section treats the theory about MOOPs and presents a class of methods to solve them.

3 MOOPs: Theory and solution methods

3.1 Problem statement

Let $\vec{X} \in \mathbb{R}^n$ represent the vector of design parameters

$$\vec{X} = (x_1, \ldots, x_n)'$$

$\mathbb{R}^n$ will be called the input space or parameter space. Similarly, let $\vec{f}$ be a set of k non-negative design objectives

$$f_i(\vec{X}) \geq 0$$

to be minimized 'simultaneously' (the meaning of it will be clarified below), then

$$\vec{F} = (f_1(\vec{X}), \ldots, f_k(\vec{X}))'$$

is in $\mathbb{R}^k$, the output space or performance space. $\vec{f}$ is a mapping from the input space $\mathbb{R}^n$ to the output space $\mathbb{R}^k$, generically non-linear, because the design objectives f_i are almost always non-linear functions of the design parameters x_j.
Furthermore, if the set of feasible design parameters is denoted by $\mathcal{X} \subseteq \mathbb{R}^n$, the set of attainable outputs is given by

$$\mathcal{F} = \left\{ \vec{F} = \vec{f}(\vec{X}) \mid \vec{X} \in \mathcal{X} \right\}$$

Since $\mathcal{F} \subset \mathbb{R}^k$, $\mathcal{F}$ has a boundary $\partial\mathcal{F}$, of special interest to the designer, since it forms a barrier in the performance space.

The MOOP can now be stated as

$$\min_{\vec{X} \in \mathcal{X}} \quad \vec{f}(\vec{X}) \tag{1}$$

We want to minimize simultaneously all design objectives f_i, subject to the given constraints. Since possibly some of the objectives are competing with one another, we need to be more specific about what we mean by 'simultaneous minimization'.

3.2 Non-inferior points

The most commonly accepted notion for 'optimal' solutions of MOOPs is the concept of non-inferiority.

Definition 1 *A design $\vec{X}^*$ is said to be non-inferior, if there are no other $\vec{X} \in \mathcal{X}$ such that*

$$\forall i \;:\; f_i(\vec{X}) \leq f_i(\vec{X}^*), \quad \textit{and} \quad \exists j \;:\; f_j(\vec{X}) < f_j(\vec{X}^*) \tag{2}$$

A design $\vec{X}^*$ for which there exist designs $\vec{X} \in \mathcal{X}$ that satisfy condition(2), is said to be inferior, because some f_j can be decreased, without increasing the remaining f_i.

From Definition 1, it follows that a non-inferior design corresponds to an exact performance trade-off. In practice, a non-inferior design is one for which an improvement in one of the objectives leads to a degradation in at least one of the remaining objectives. Since there are many designs that satisfy these conditions, it is evident to think about a set of 'optimal' solutions, instead of looking for a single solution.

Definition 2 *If the set of all non-inferior designs is denoted*[1] *by $\mathcal{X}_p$, then the set of non-inferior performances is given by*

$$\mathcal{F}_p = \left\{ \vec{F} = \vec{f}(\vec{X}) \mid \vec{X} \in \mathcal{X}_p \right\}$$

Note that the set $\mathcal{F}_p$ contains all the performances on the trade-off boundary. Since we are interested in finding designs with performance on the trade-off boundary, the ultimate challenge could be to find all these designs. The problem could then be restated as:

> *Find the set of non-inferior designs $\mathcal{X}_p$, corresponding to the set of non-inferior performances $\mathcal{F}_p$.*

However, this problem will almost never be solved completely. In general, the problem to solve reduces to:

---

[1] Non-inferior designs are also known as Pareto-optimal designs, referring to the work done by Pareto in 1896 in the context of economic utility theory [13]. This explains the index p in $\mathcal{X}_p$ and $\mathcal{F}_p$

Find a non-inferior design $\vec{X} \in \mathcal{X}_p$, corresponding to an 'acceptable' non-inferior performance $\vec{F} \in \mathcal{F}_p$.

The following example (see also figure 2) illustrates the introduced concept and the definitions.

Example 1 *Consider a MOOP with 2 objectives and 4 constraints:*

$$\min_{(x_1,x_2)} \left(x_1^6 + x_2^2,\ \frac{1}{x_1^2} + x_2^2\right),\ s.t.\ 0 \le x_1 \le 1.5,\ 0 \le x_2 \le 1$$

The set of feasible parameters $\mathcal{X}$ is

$$\mathcal{X} = \{(x_1, x_2) \mid 0 \le x_1 \le 1.5,\ 0 \le x_2 \le 1\}$$

The set of attainable performances $\mathcal{F}$ is the image of $\mathcal{X}$, for the mapping

$$\vec{f} = \left(x_1^6 + x_2^2,\ \frac{1}{x_1^2} + x_2^2\right)$$

It is easy to verify that the set of non-inferior designs is given by

$$\mathcal{X}_p = \{(x_1, x_2) \mid (x_1, x_2) \in \mathcal{X} \quad and \quad x_2 = 0\},$$

while the set of non-inferior performances is given by

$$\mathcal{F}_p = \left\{(f_1, f_2) \mid f_1 = x_1^6,\ f_2 = \frac{1}{x_1^2},\ (x_1, x_2) \in \mathcal{X}\right\}$$

This implies that the trade-off curve can be written as

$$f_2 = \frac{1}{f_1^{\frac{1}{3}}}$$

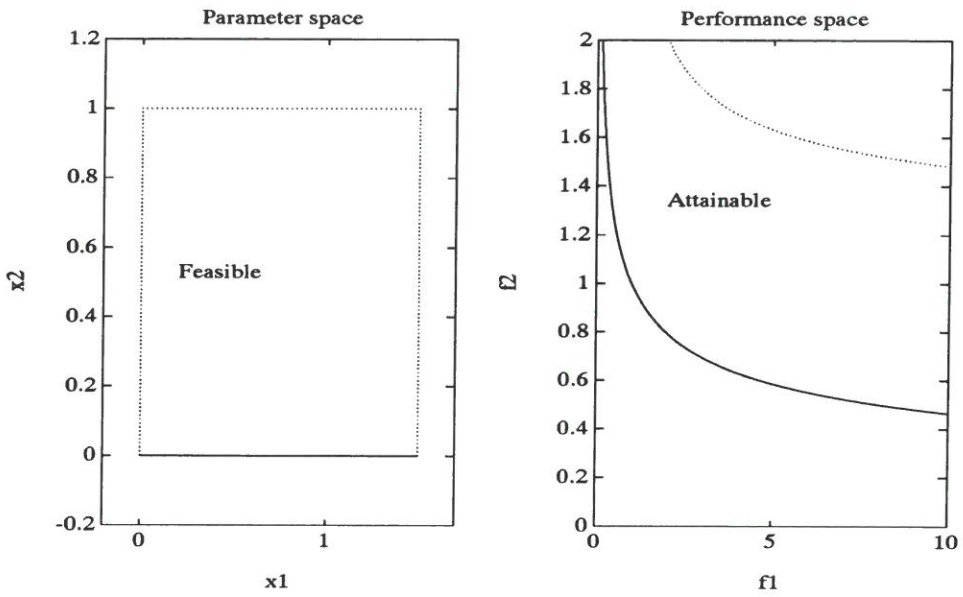


Figure 2: $\vec{f} = (f_1, f_2)$ *maps the set of feasible designs* $\mathcal{X}$ *on the set of attainable designs* $\mathcal{F}$ *(dotted lines), and it maps the set of non-inferior designs* $\mathcal{X}_p$ *on the set of non-inferior performances* $\mathcal{F}_p$ *(full lines). Although in this example the set* $\mathcal{X}_p$ *coincides with a part of the boundary of* $\mathcal{X}$*, this is generically not the case. Note that* $\mathcal{F}_p$ *contains all the performance trade-offs of our interest.*

3.3 Solving the MOOP

In the literature, a variety of techniques is available for finding non-inferior solutions. Although the method of inequalities [21], hierarchial optimization [19] and interval elimination methods [14] perform quite well, we only use methods based on function scalarization. First, we show how these methods transform the vector minimization problem into a function minimization problem. Next, we present a non-linear programming algorithm to solve the transformed problems.

3.3.1 Transforming the MOOP

Before presenting the different methods, we emphasize that no unique solution exists. In fact, there exists a set of non-inferior solutions (see definitions 1 and 2). In general, since we are interested in finding a design with an 'acceptable' performance trade-off, it is reasonable to look for a technique that allows us to pick out the non-inferior design with the non-inferior performance that corresponds to the performance trade-off we intend.

Depending on the chosen scalarization method, different techniques are used. In general, the user is asked to set some parameters to express the 'desired' performance trade-off. The obtained non-inferior solution varies with these parameters.

Vector norm optimization

In this approach, the obtained solution is determined by putting weights before the objectives. If the weights are denoted by λ_i, then the weighted problem is stated as:

$$\min_{\vec{X}\in\mathcal{X}} \quad \vec{f}_\lambda(\vec{X}) = (\lambda_1 f_1(\vec{X}), \ldots, \lambda_k f_k(\vec{X}))$$

An often used technique to minimize all components of a vector $\vec{Y} = (y_1, \ldots, y_k)$ is to minimize a certain vector norm. It can be found in [8] that the conditions to be a norm are satisfied by the p-norm, defined as

$$\| \vec{Y} \|_p = \left(\sum_{i=1}^{k} |y_i|^p \right)^{\frac{1}{p}}$$

where p is assumed to be a strictly positive integer. Using this norm, the transformed problem becomes

$$\min_{\vec{X}\in\mathcal{X}} \quad \| \vec{f}_\lambda(\vec{X}) \|_p$$

Note that for $p = 1$, the method reduces to the weighted-sum method [3], while for $p = \infty$, it reduces to the minimax approach found in [17]. A geometrical interpretation, related to example 1, is given in figure 3, for $p = 1$, $p = 2$ and $p = \infty$.
Different solutions for problem 1 can be obtained by varying the weights. However, it should be noticed that problems arise for non-convex sets $\mathcal{F}_p$. Then, only large values of p guarantee that all non-inferior points are found by varying the weights. An intuitive explanation could be that for increasing values of p, the contour lines of the p-norm of the weighted vector become more sharp (in the direction indicated by the weights), which implies that all non-inferior solutions can be obtained by varying the weights.

The ϵ constraint method
In this approach, the designer chooses one objective to be minimized, while the other objectives are required to be smaller then a certain value ϵ_j. The obtained solution is obviously determined by the values of ϵ_j. The transformed problem becomes

$$\min_{\vec{X}\in\mathcal{X}} \quad f_i(\vec{X}) \ s.t. \ f_j \leq \epsilon_j, \ \forall j \neq i$$

A geometrical interpretation, related to example 1, is given in figure 4. It should be noticed that all solutions of problem 1 are obtained, when the values of ϵ_j are varied appropriately [2]. A bad choice of ϵ_j leads to an infeasible problem.

The goal attainment method
In the goal attainment method, the obtained solution is determined by parameters which are meaningful to the designer. Let $\vec{f}^* = (f_1, \ldots, f_k)$ be the designer's preferred solution (the performance goal) and let $\vec{\lambda}$ be the desired search direction towards this goal, then the transformed problem becomes

$$\min \quad \gamma \ s.t. \ \vec{X} \in \mathcal{X}, \ f_i(\vec{X}) - \lambda_i \gamma \leq f_i^*, \ \forall i$$

A geometrical interpretation, related to example 1, is given in figure 5.
Note that for $\gamma < 0$, the designer's wishes are overattained, which means that the performance of the obtained design is higher than the designer's goal, while for $\gamma > 0$, the designer's wishes are underattained, which means that the goal was unrealistic, thus not in $\mathcal{F}$, the set of attainable objectives [6].
By varying the goal $\vec{f}^*$ and the direction $\vec{\lambda}$, it is possible to obtain all solutions to problem 1.

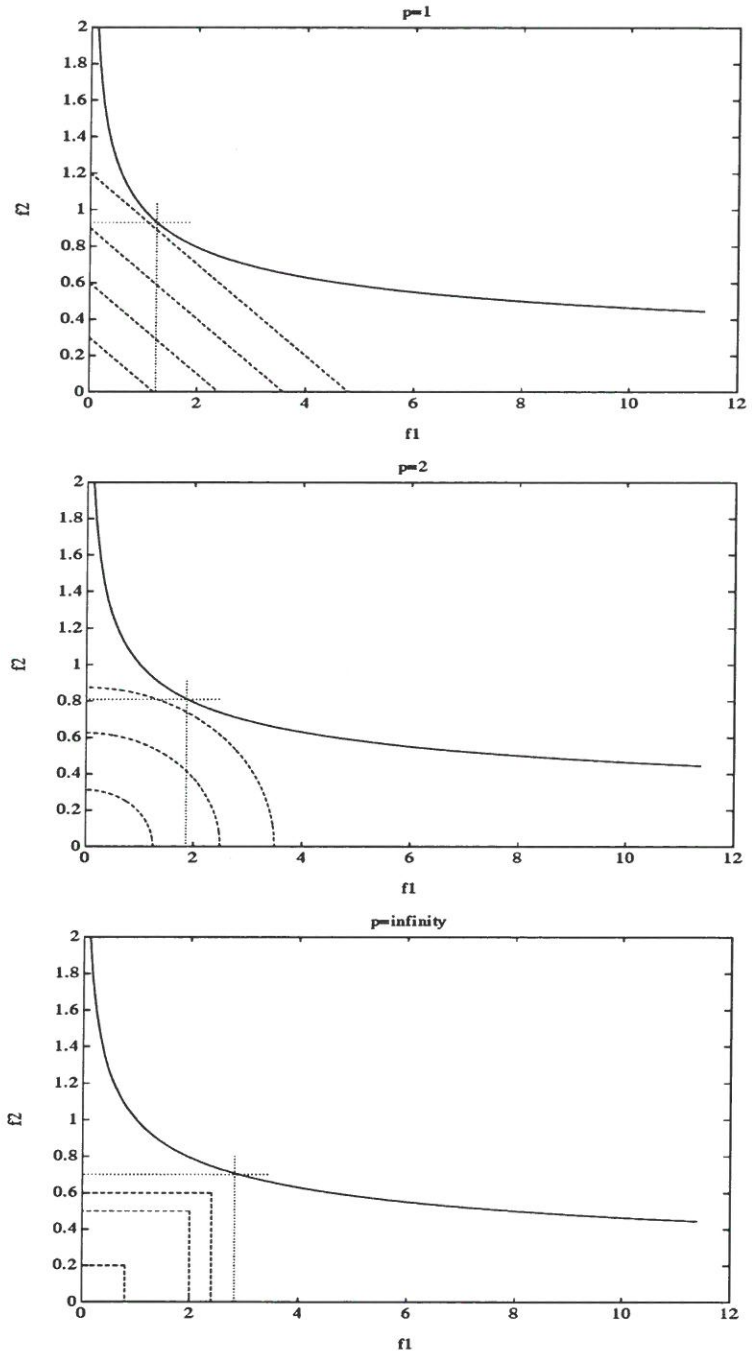


Figure 3: *The norm minimization method for example 1. The full line represents the trade-off curve to be found, the dashed lines are the contour levels of the vector norms. The weighting vector* $\vec{\lambda}$ *is* $(0.2, 0.8)$. *(1)* $p = 1$, *the obtained design is* $(x_1, x_2) = (1.036, 0)$, *corresponding to a performance* $(f_1, f_2) = (1.24, 0.93)$. *(2)* $p = 2$, *the obtained design is* $(x_1, x_2) = (1.110, 0)$, *corresponding to a performance* $(f_1, f_2) = (1.87, 0.81)$. *(3)* $p = \infty$, *the obtained design is* $(x_1, x_2) = (1.189, 0)$, *corresponding to a performance* $(f_1, f_2) = (2.83, 0.70)$.

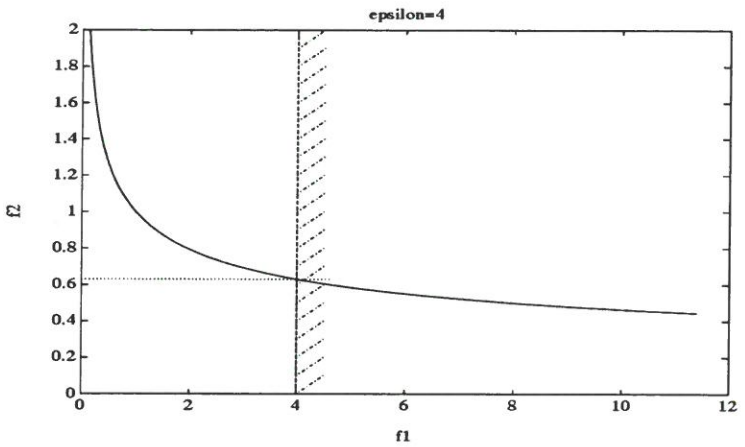


Figure 4: *The ϵ constraint method for example 1. The full line represents the trade-off curve to be found, the dashed line indicates that f_1 must be smaller than $\epsilon_1 = 4$. The obtained design is $(x_1, x_2) = (1.260, 0)$, corresponding to a performance $(f_1, f_2) = (4, 0.63)$.*

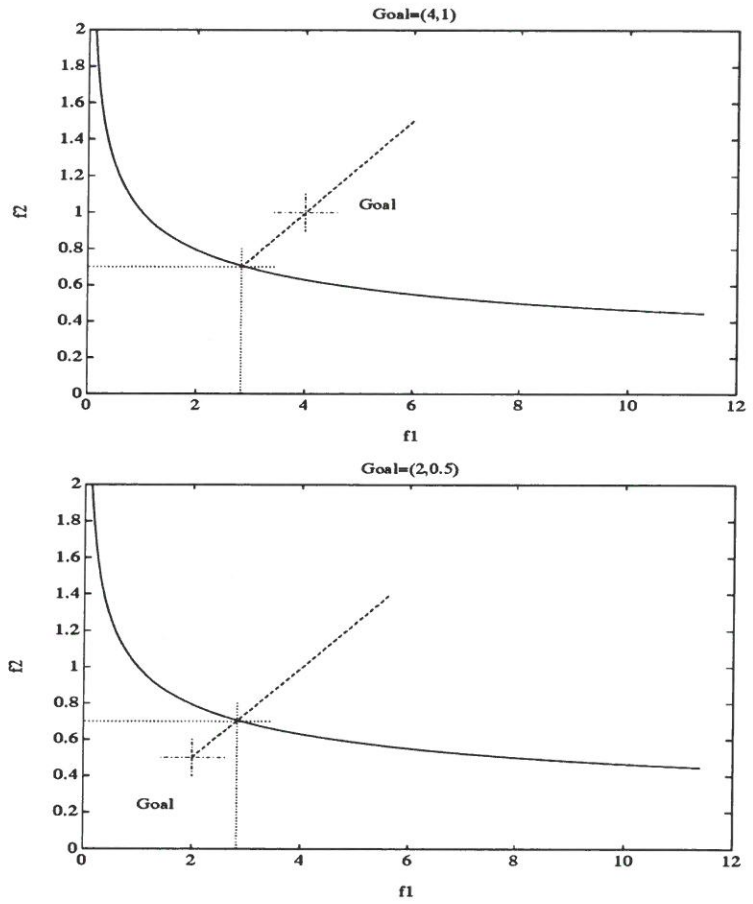


Figure 5: *The goal attainment method for example 1. The weighting vector is* $\vec{\lambda} = (4, 1)$. *The full line represents the trade-off curve to be found, the dashdotted line is the line along which* γ *varies. (1) Goal=*$(4, 1)$, *the obtained design is* $(x_1, x_2) = (1.189, 0)$, *with corresponding performance* $(f_1, f_2) = (2.83, 0.70)$ *and* $\gamma = -0.2925 < 0$. *The design is better than expected. (2) Goal=*$(2, 0.5)$, *the obtained design is* $(x_1, x_2) = (1.189, 0)$, *with corresponding performance* $(f_1, f_2) = (2.83, 0.70)$ *and* $\gamma = 0.2075 > 0$. *The design is worse than expected.*

3.3.2 Solving the non-linear programming problem

In this paper, three different methods are proposed to transform the problem (finding one solution of the MOOP (1)) into a function minimization problem. The general problem that shows up, is a non-linear programming problem, stated as

$$\min_{\vec{X}} \; f(\vec{X}) \; s.t. \; \vec{g}(\vec{X}) = (g_1(\vec{X}), \ldots, g_l(\vec{X})) \geq 0 \tag{3}$$

To solve problem (3), the technique of sequential programming is used with a logarithmic barrier function, as proposed in [5]. Then, the problem reduces to

$$\min_{\vec{X}} \; L(\vec{X}, r) = f(\vec{X}) - r \sum_{i=1}^{l} \log(g_i(\vec{X}))$$

where r is a positive scalar.

The idea behind this barrier function is the following: because $\log(g_i(\vec{X}))$ becomes $-\infty$ for values of $g_i(\vec{X}) \to 0$, the function $L(\vec{X}, r)$ increases when the boundaries of the feasible region are reached. Since the algorithm tries to minimize $L(\vec{X}, r)$, it can be seen intuitively that no infeasible solutions will be generated by the minimization algorithm. It should be noticed that a smaller value of r gives more importance to the minimization of f.

Initially, the minimization is done for a large value of r, to obtain a feasible solution. Next, the value of r is decreased and the minimization is done again, with the previous minimum as starting point. In this way, it is possible to construct a path of minimizers $\vec{X}^*(r)$. It is clear that

$$\vec{X}^* = \lim_{r \to 0} \vec{X}^*(r)$$

is a solution of the non-linear programming problem 3.

Because it is impossible to solve the problem for all values of r, a finite sequence of decreasing values r_k will be considered. A difficulty is the choice of the sequence r_k. Although theoretical results exist for convex problems [16], no rigourous methods exist for the general problem. A practical formula to generate the sequence r_k is $r_k = \frac{r_{k-1}}{\alpha}$, with $\alpha > 1$. For small values of α, continuity and convergence of the path $\vec{X}^*(r)$ are guaranteed, but the time to converge is very long. Oppositely, large values of α are able to reduce this time, but might result in a discontinuous, non-converging path. Other rules to generate the sequence of r-values can be found in [5].

An important property of the logarithmic barrier function is that $L(\vec{X}, r)$ is convex in $\vec{X}$, if $f(\vec{X})$ is convex and all $g_i(\vec{X})$ are concave in $\vec{X}$.

The following non-linear programming problem (see also figure 6) illustrates the technique of sequential programming with a logarithmic barrier function.

Example 2 *The problem to be solved is stated as*

$$\min_{(x_1,x_2)} \; x_1 + x_2 \;\; s.t. \;\; (-x_1^2 + x_2, x_1) \geq 0$$

With a logaritmic barrier function, the problem reduces to

$$\min_{(x_1,x_2)} \; L(x_1, x_2, r) = x_1 + x_2 - r\log(-x_1^2 + x_2) - r\log(x_1)$$

for a decreasing sequence of r-values. When the minimizer $\vec{X}^*(r = 0)$ is obtained, the problem is solved. In practice, this minimizer is estimated, by extrapolating the path of minimizers $\vec{X}^*(r \to 0)$.

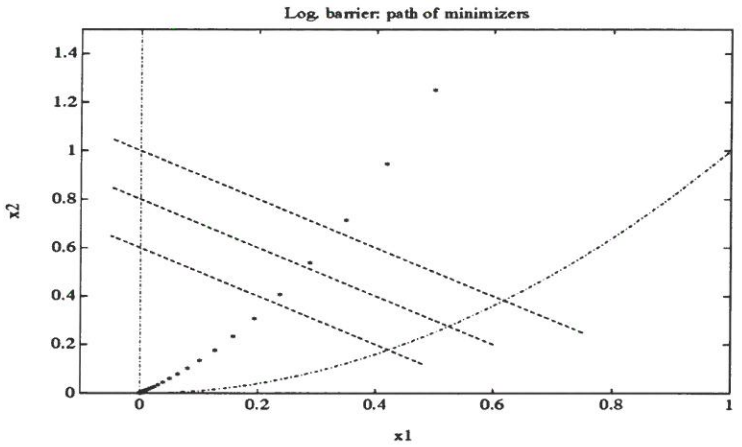


Figure 6: *The logaritmic barrier method, applied to example 2. The sequence of decreasing r-values is generated using the formula $r_k = \frac{r_{k-1}}{\alpha}$, with $r_0 = 1$ and $\alpha = 1.3$. The path of minimizers $X^*(r_k)$ is marked by stars (*). The constraints are indicated by dashdotted lines, while the dashed lines give the contour levels of the objective function $x_1 + x_2$.*

4 A numerical example

Consider the theoretical problem [15] of designing a dynamical control law $K(s)$ for the following one-degree-of-freedom (1DOF) configuration (see figure 7).

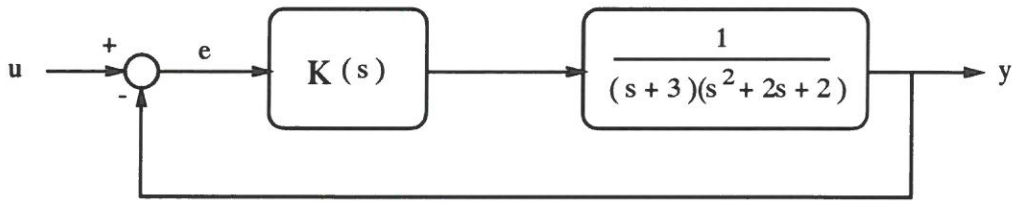


Figure 7: *A 1DOF configuration with a control law $K(s)$ to be designed*

The first **parametrization** to investigate is a classical PI-control law. In that case, the controller may be parametrized as

$$K(s) = P + \frac{I}{s}$$

which means that the vector $\vec{X}$ is given by $(x_1, x_2) = (P, I)$.

The **design specifications** of interest are:

- As hard specifications (to be satisfied always)
  - Realizability

    For the case of PI-control, no explicit condition on the parameters needs to be added.
  - Stability

    From the Routh-Hurwitz stability test, it follows that the following conditions need to be satisfied:,

    $$\Phi_{hard,1} = x_2 \geq 0, \quad \Phi_{hard,2} = 33 - x_1 \geq 0$$

    $$\Phi_{hard,3} = (33 - x_1)(7 + x_1) - 25x_2 \geq 0$$
  - Bounds on the design variables

    $$\Phi_{hard,4} = 100 - x_1 \geq 0, \quad \Phi_{hard,5} = 100 - x_2 \geq 0$$

    $$\Phi_{hard,6} = 100 + x_1 \geq 0, \quad \Phi_{hard,7} = 100 + x_2 \geq 0$$
  - Zero steady state error for a step input

    No additional condition on the parameters is required, since the integral term in $K(s)$ guarantees a zero steady state error for step inputs.
- As soft specifications (to be traded-off)
  - The overshoot of the unit step response $y(t)$. This measure is computed as

    $$\Phi_{soft,1} = \max\{\max(y(t) - 1), 0\}$$
  - The l_2-norm of the error signal $e(t)$ for a unit step input $u(t)$, a measure for the energy in the error signal, is also a measure for the tracking behaviour. It is computed as

    $$\Phi_{soft,2} = \left(\int_0^\infty e^2(t)dt\right)^{\frac{1}{2}}$$

The MOOP can then be stated as

$$\min_{(x_1,x_2)} (\Phi_{soft,1}, \Phi_{soft,2}) \ \ s.t. \quad (\Phi_{hard,1}, \ldots, \Phi_{hard,7}) \geq 0$$

This problem is then transformed into a scalar minimization problem using the method of vector norm minimization.

If we choose $p = 1$, the problem becomes

$$\min_{\vec{X}} \ \lambda_1 \Phi_{soft,1}(\vec{X}) + \lambda_2 \Phi_{soft,2}(\vec{X}) \quad \text{s.t.} \ (\Phi_{hard,1}, \ldots, \Phi_{hard,7}) \geq 0$$

with $\Phi_{soft,1}$ measuring the overshoot and $\Phi_{soft,2}$ as a measure for the energy. By varying the weights, we try to find different non-inferior solutions on the trade-off curve $\mathcal{F}_p$. As initial design, we use $(x_1, x_2) = (5, 10)$. The results for various weights can be found in the following table. As usual, the overshoot is expressed in %. Note that high values of λ_1 ask for a small overshoot, while high values of λ_2 ask for low energy.

(λ_1, λ_2)	(x_1, x_2)	(Φ_1, Φ_2)
$(0.95, 0.05)$	$(9.49, 2.26)$	$(0.0, 1.00)$
$(0.5, 0.5)$	$(8.40, 3.19)$	$(2.3, 0.94)$
$(0, 1)$	$(11.23, 4.69)$	$(20.8, 0.84)$

Figure 8 shows the corresponding step responses, while figure 9 shows the trade-off curve for a PI-controller.
It should be noticed that the obtained trade-off curve is in fact an upper approximation of the exact one. However, an evaluation of grid points over the considered parameter box showed that the obtained curve was a very good approximation of the exact curve.

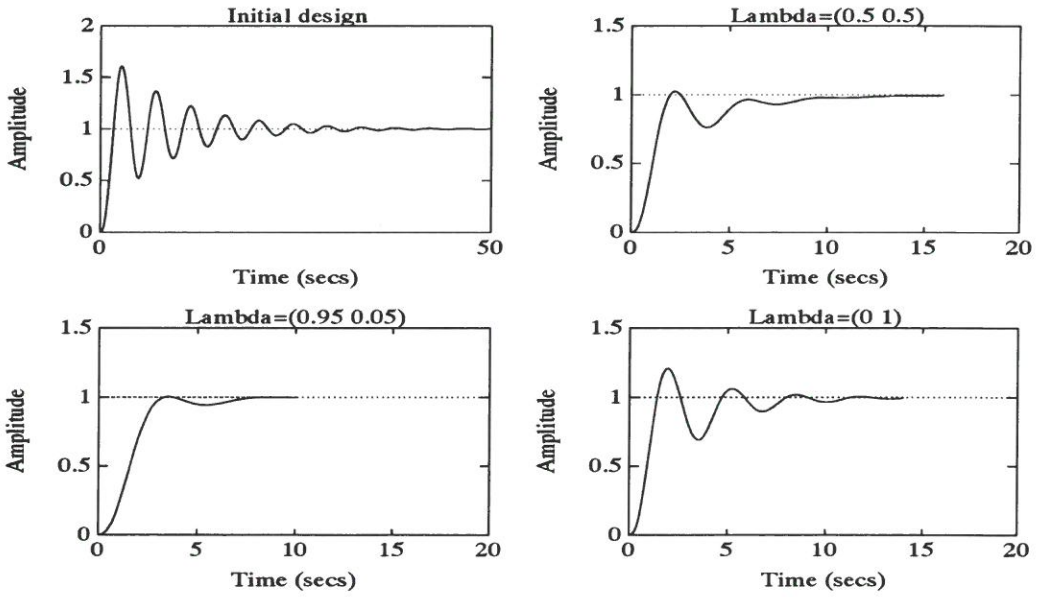


Figure 8: *Step responses with a PI-controller: the initial design and non-inferior designs for various weights*

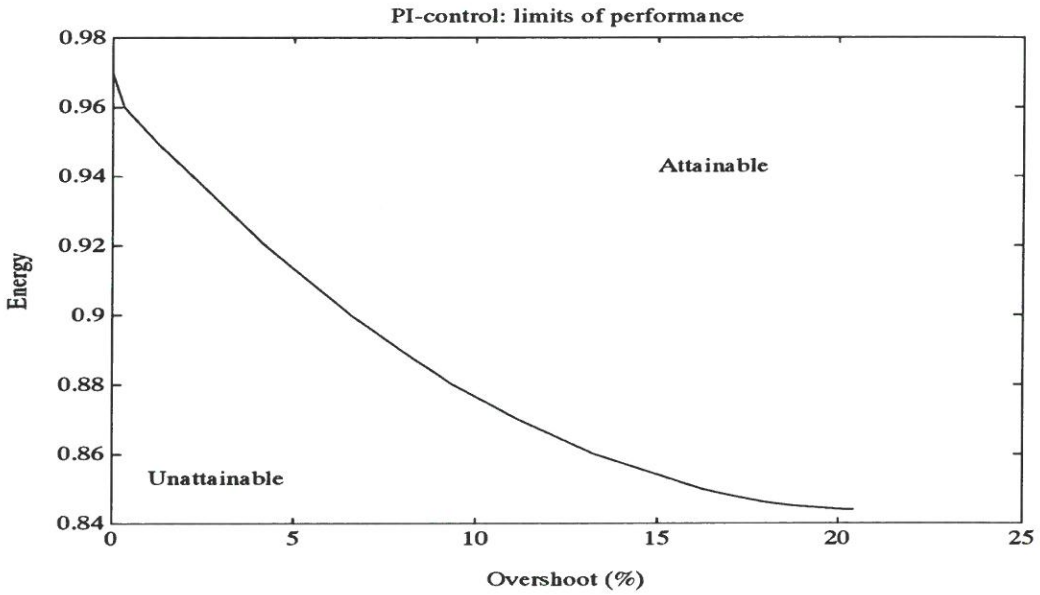


Figure 9: *The trade-off curve for the classical PI-parametrization.*

The same problem is also solved for some other parametrizations:

- Two other classical parametrizations of the control law $K(s)$.
  - A compensator network
  - A PID-controller
- A modern controller, based on the Youla-parametrization. Instead of designing the control law $K(s)$ directly, the Youla parameter $Q(s)$ is designed first. Once $Q(s)$ is given, the control law $K(s)$ can readily be computed [7].

The following table shows the different controllers, the parametrizations, the unknowns and the initial designs. To be able to compare the different types of control, the results for the PI-parametrization are included also.

Controller	Parametrization	Unknowns	Initial design
PI-controller	$K(s) = P + \frac{I}{s}$	$\vec{X} = (P, I)$	$(5, 10)$
Compensator network	$K(s) = K\frac{s-\alpha}{s(s+\alpha)}$	$\vec{X} = (K, \alpha)$	$(-5, 5)$
PID-controller	$K(s) = P + \frac{I}{s} + Ds$	$\vec{X} = (P, I, D)$	$(5, 10, 10)$
Modern controller	$Q(s) = \frac{\sum_{i=0}^{5} x_i s^i}{(s+2)(s+3)(s+4)(s+5)^2}$	$\vec{X} = (x_0, \ldots, x_5)$	$(6, 1, 1, 1, 1, 1)$

The resulting objective values (Φ_1, Φ_2) can be found in the following table, for different weight values.

Name	$\vec{\lambda} = (0.95, 0.05)$	$\vec{\lambda} = (0.5, 0.5)$	$\vec{\lambda} = (0, 1)$
PI-controller	$(0.0, 1.00)$	$(2.3, 0.94)$	$(20.8, 0.84)$
Compensator network	$(0.0, 2.54)$	$(19.5, 2.02)$	$(29.3, 1.98)$
PID-controller	$(0.0, 0.39)$	$(3.0, 0.23)$	$(40.1, 0.13)$
Modern controller	$(0.1, 0.93)$	$(0.7, 0.22)$	$(18.9, 0.19)$

Figure 10 shows the trade-off curves for the different parametrizations.
It is clear that if the trade-off curves can be computed for different control laws, then these curves can be used to decide not only on parameters, but also on the control laws themselves! In this example, a PID-law turns out to be the best classical controller. Depending on the weight values, the modern controller performs better or worse than the best classical controller (figure 10).

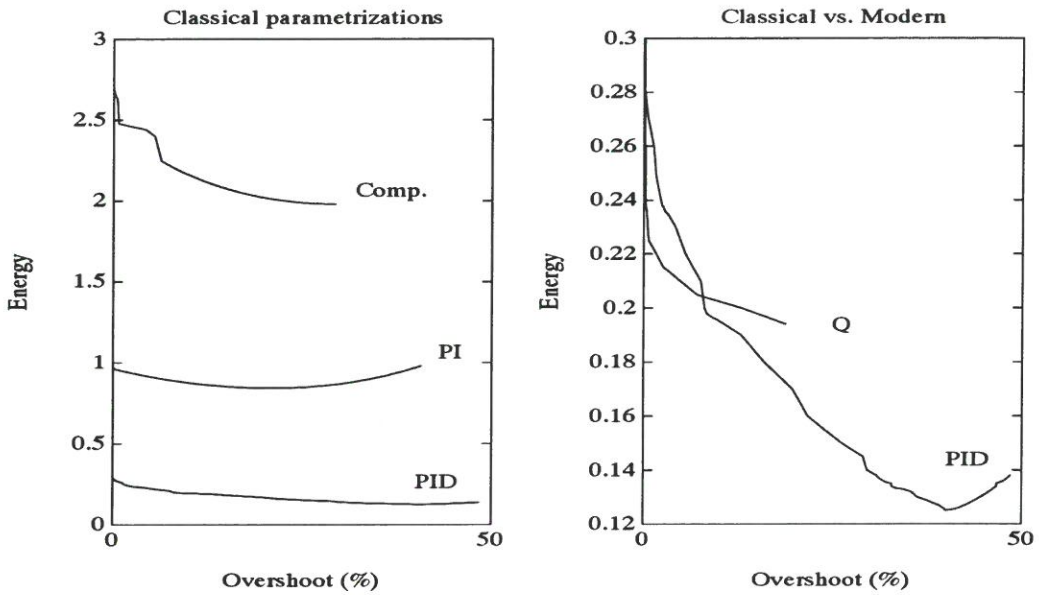


Figure 10: *(1) Trade-off curves for the classical parametrizations. For a certain type of controller, only the performances above the trade-off curve are attainable. (2) Classical versus modern parametrization. Because a Q-parameter with a finite order is designed, the classical PID controller performs better than the modern controller for some weight values!*

5 Some notes about the design time?

In Section 2, it is explained how a practical design problem is translated into a MOOP, while Section 2 gives a class of methods to solve the MOOP. As stated in Section 3, no unique solution exists. Although it is possible for the designer to influence the solution that will be obtained, it should be clear that the 'ideal' solution is almost never found immediately. In practice, this means that the time to find an acceptable design is not equal, but proportional to the time to find one single solution.
To make the approach useful for practical applications, this design time should be reduced as far as possible.
Although many factors influence the time to solve the non-linear programming problem in Section 3, three of them are extremely important. In the following subsections, they are discussed in the context of system and control design.

5.1 The nature of the problem: convex or non-convex ?

The non-linear programming problem 3 is said to be convex, if $f(\vec{X})$ is convex in $\vec{X}$ and $g_i(\vec{X})$ is concave in $\vec{X}$ for all i. Note that the proposed logarithmic barrier function doesn't affect the convexity of the problem: it only transforms a convex constrained optimization problem into a convex unconstrained optimization problem.

If convexity can be guaranteed, then some interesting facts should be noticed:

- For convex problems, very efficient algorithms are described in literature. The ellipsoid method of Yudin, Shor and Nemirovsky [12] and Kelley's cutting plane method [11] are space reduction algorithms, while Nesterov's and Nemyrovsky's polynomial time methods are sequential search algorithms based on logarithmic barrier functions [16]. For all these methods, fast convergence can be proven.

- If the problem is convex, then the feasible parameter region $\mathcal{X}$ and the attainable performance region $\mathcal{F}$ are convex also [10]. In that case, convex inner and outer approximations of $\mathcal{X}$ and $\mathcal{F}$ can be used without much loss of accuracy.

  In the context of trade-off calculation in dynamical design, this means for $\mathcal{X}$ that almost all feasible designs can be described using a convex combination of a few designs on the boundary (design nodes). For $\mathcal{F}$, it means that only a low number of performances on the surface of non-inferior points (performance nodes) needs to be calculated to describe the trade-off surface accurately (see figure 11).

A good example of a convexity approach for dynamical design is given by Boyd and Baratt in [2], where the controller design problem for MIMO linear plants is treated as a closed loop convex problem.

5.2 Local low order approximations: available or estimated?

Most algorithms to minimize a multivariable function use local low order approximations to decide upon the new search direction to be taken. First order methods, like steepest descent, use only first order information (the gradient or sometimes the subgradient), while second order methods, like the Newton methods, use not only the gradient, but also the Hessian to compute the new search direction.

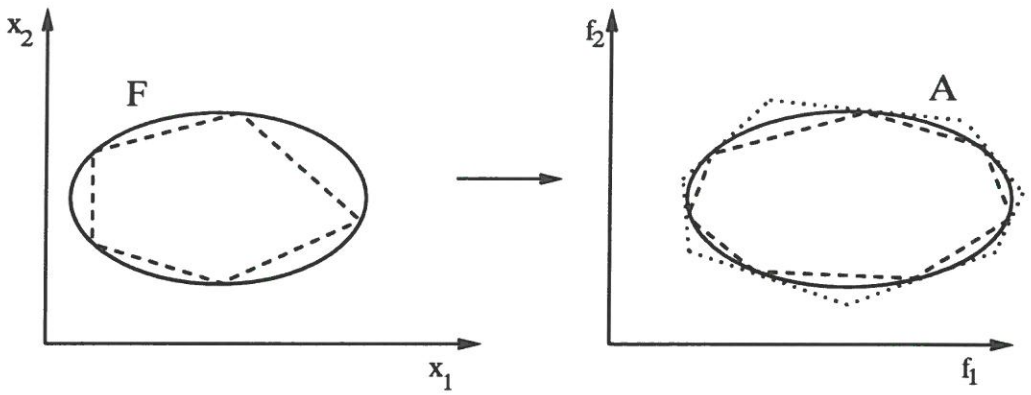


Figure 11: *Convex approximations for $\mathcal{X}$ and $\mathcal{F}$. The dashed lines form inner approximations, the dotted lines form an outer approximation.*

Experimentally, it is easy to verify that all methods perform better and faster with exact higher order information, than with estimated information calculated by finite differences.
In dynamical design, exact higher order information can only be computed straightforward for problems treating linear systems with an appropriate parametrization. A good example is given in [18], where formulas for the first and second derivatives to the elements of the state space matrices are derived, for step responses, polynomial responses and frequency responses. However, for general dynamical design problems, exact formulas are hard to find, so finite difference approximations are necessary.

5.3 The time needed for a function evaluation

Function evaluations are needed in optimization routines in mainly two different operations: to determine the optimal steplength (i.e. optimal line search or bisection) and to estimate local first order and second order approximations. Because both operations are essential for efficient optimization, it is clear that the time to evaluate a function should be as small as possible.
In general, it holds that analytical computation of a function value is faster than numerical computation. However, in dynamical design, only a small number of problems are involved with a structure, a parametrization and design specifications that allow all functions to be evaluated analytically.
In practice, in order to be able to solve design problems for a broad class of structures, parametrizations and design specifications, a simulator and a performance evaluator are implemented. This means that almost all functions are evaluated numerically.

6 Conclusions

In this paper, a general framework for a multi-objective optimization approach to parameter setting in system and control design is developed. In three different phases, the problem can be mathematically formulated as a MOOP. Next, the resulting vector minimization problem is transformed into a function minimization problem. Finally, this transformed problem is solved using a sequential programming approach with a logarithmic barrier.

An example is used to show that the method finds 'optimal' solutions. If the trade-off curves can be computed for different structures, this approach allows us not only to decide on parameters, but it also gives us a tool to decide on the structures themselves.
However, to make the approach useful for applications, the design time should be reduced as far as possible. Once the parametrization and the design specifications are chosen, the design time can be 'predicted' in some sense.

References

[1] Arora J.S. *Introduction to optimum design.* Mc Graw-Hill, New York, 1989.

[2] Boyd S.P., Baratt C. *Linear Controller Design, Limits of performance.* Prentice Hall Inc., 1990.

[3] Brayton R.K., Spence R. *CAD of electrical circuits, 2, Sensitivity and Optimization.* Elsevier Scientific Publishing Company, 1980.

[4] Dorf R.C. *Modern Control Systems.* Addison-Wesley, New York, 1989.

[5] Fiaco A.V., Mc Cormick G.P. *Non-linear programming; Sequential Unconstrained Minimization Techniques.* John Wiley and Sons, 1968.

[6] Fleming P.J., Pashkevich A.P. *Computer-Aided Control System Design Using a Multi-Objective Optimization Approach.* Control '85 Conference, Cambridge, U.K., 174-179, 1985.

[7] Francis B.A. *A Course in H_∞ Control.* Springer-Verlag, 1988.

[8] Golub G., Van Loan C. *Matrix Computations.* Johns Hopkins Univ. Press, Second edition, 1989.

[9] Grace A.C.W. *Computer-Aided Control System Design Using Optimization Techniques.* PhD. thesis, University of Wales, Bangor, Gwynedd, UK, 1989.

[10] Jongen H.T. *Grondslagen optimalisatie theorie.* Lecture Notes, University of Twente, Enschede, 1987.

[11] Kelley J.E. *The cutting plane method for solving convex programs.* J. Soc. Ind. Appl. Math., Vol. 8, No. 4, 703-712, December 1960..

[12] Khachiyan L. *A polynomial algorithm in linear programming.* Soviet Math. Doklady, 20, 191-194, 1979.

[13] Pareto V. *Cours d' Economie Politique.* Lausanne, Rouge, 1896.

[14] Payne A.N., Polak E. *An interactive rectangle elimination method for bi-objective decision making.* IEEE Trans. on Automatic Control, AC-25, 421-432, 1980.

[15] Polak E., Mayne D.Q. *An Algorithm for Optimization Problems with Functional Inequality Constraints.* IEEE Trans. on Automatic Control, AC-21, No. 2, 184-193, 1976.

[16] Nesterov J.E., Nemyrovsky A.S. *Self-concordant functions and polynomial-time methods in convex programming.* Report, Central Econ. and Math. Institute, USSR Acad. Sci., Moscow, USSR, 1989.

[17] Nye W.T., Tits L.T. *An application-oriented, optimization-based methodology for interactive design of engineering systems.* Int. J. Control, Vol. 43, No. 6, 1693-1721, 1986.

[18] Roelofs H.H.B. *Tolerance Optimization of Servomechanisms.* PhD. thesis, University of Twente, 1991.

[19] Toivonen H.T. *A multiobjective linear quadratic Gaussian control problem.* IEEE Trans. on Automatic Control, AC-29, 279-280, 1984.

[20] Vanvuchelen P., De Moor B. *Analysis of electronically controlled hydraulic servovalves using a simulation approach.* To be presented at the 26th ISATA Conference, Aachen, 1993.

[21] Zakian V., Al-Naib U. *Design of dynamical and control systems by the method of inequalities.* Proc. IEEE, Vol. 120, No. 11, November 1973.